```
<?xml version="1.0"?>
<!ELEMENT query (reaction, reaction, query?)>
<!ATTLIST query question CDATA #REQUIRED>
<!ELEMENT reaction (setpoints | query)>
<!ELEMENT setpoints EMPTY>
<!ATTLIST setpoints points NMTOKEN #REQUIRED>
```

Fig. 1

```
<?xml version="1.0"?>
<!DOCTYPE query SYSTEM "query.dtd">
<query question="Is Paris the Capital of France?">
        <setpoints points=10>
        <query question="Are you sure?">
                <setpoints points=0>
                        <setpoints points=1>
        </query>
</query>
```

Fig. 2

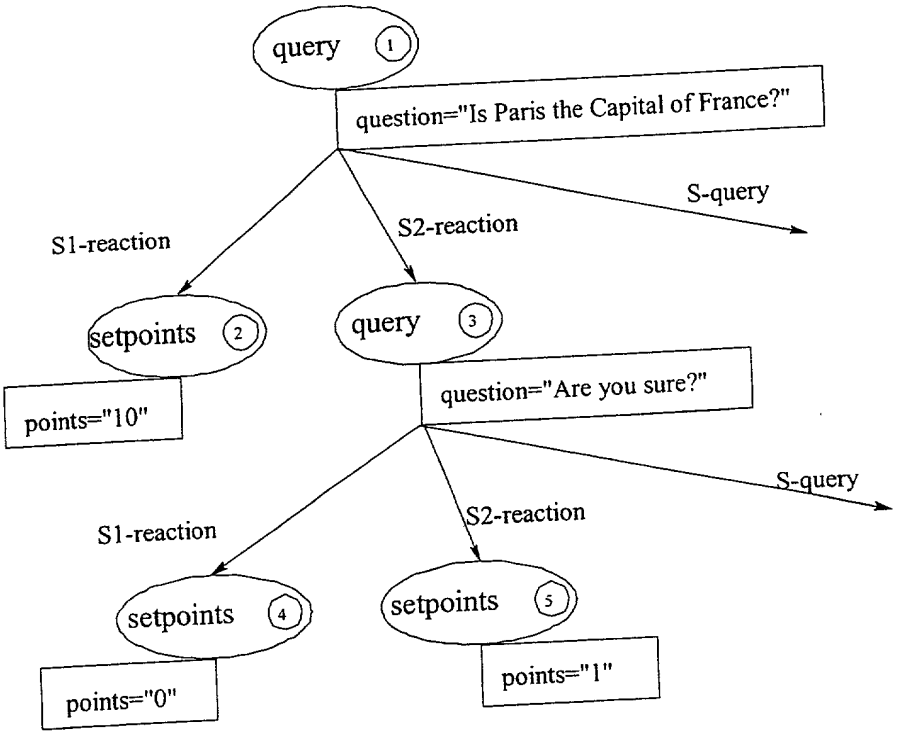

Fig. 3

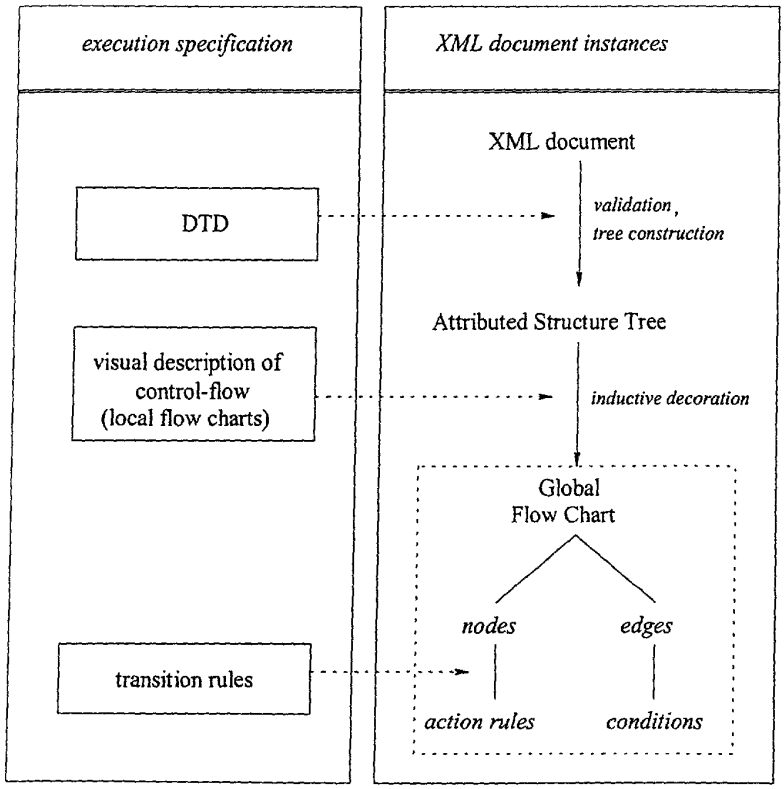
execution specification
XML document instances
XML document
DTD
validation,
tree construction
Attributed Structure Tree
visual description of
control-flow
(local flow charts)
inductive decoration
Global
Flow Chart
nodes
edges
transition rules
action rules
conditions

Fig. 4

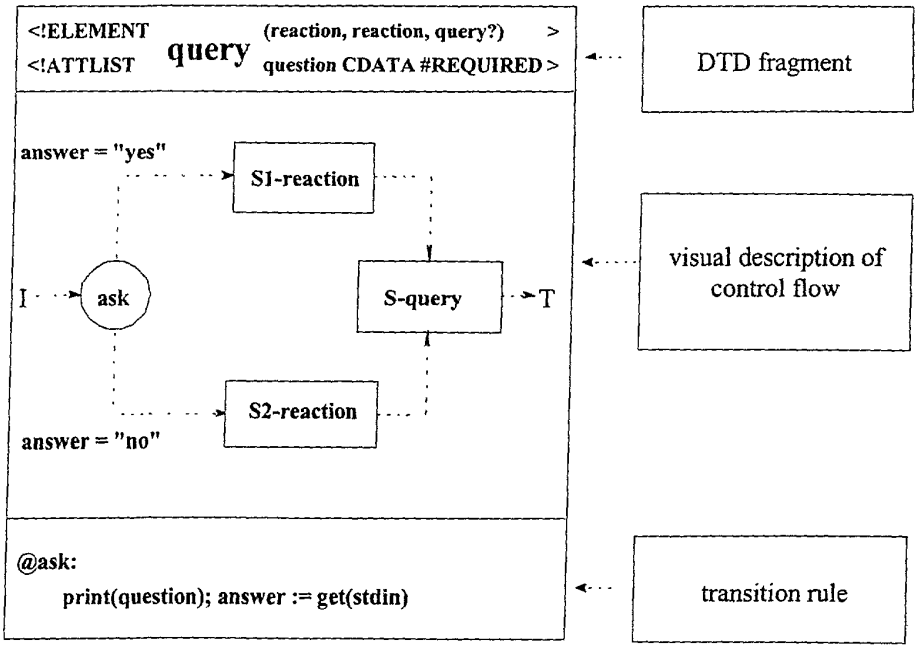
<!ELEMENT query (reaction, reaction, query?) >
<!ATTLIST question CDATA #REQUIRED >
DTD fragment
answer = "yes"
S1-reaction
visual description of
control flow
I
ask
S-query
T
S2-reaction
answer = "no"
@ask:
print(question); answer := get(stdin)
transition rule

Fig. 5

Fig 69

```
<!ELEMENT reaction (setpoints | query) >
<!ATTLIST                              >
```

Fig 66

```
<!ELEMENT setpoints                                    >
<!ATTLIST          points NMTOKEN #REQUIRED
```

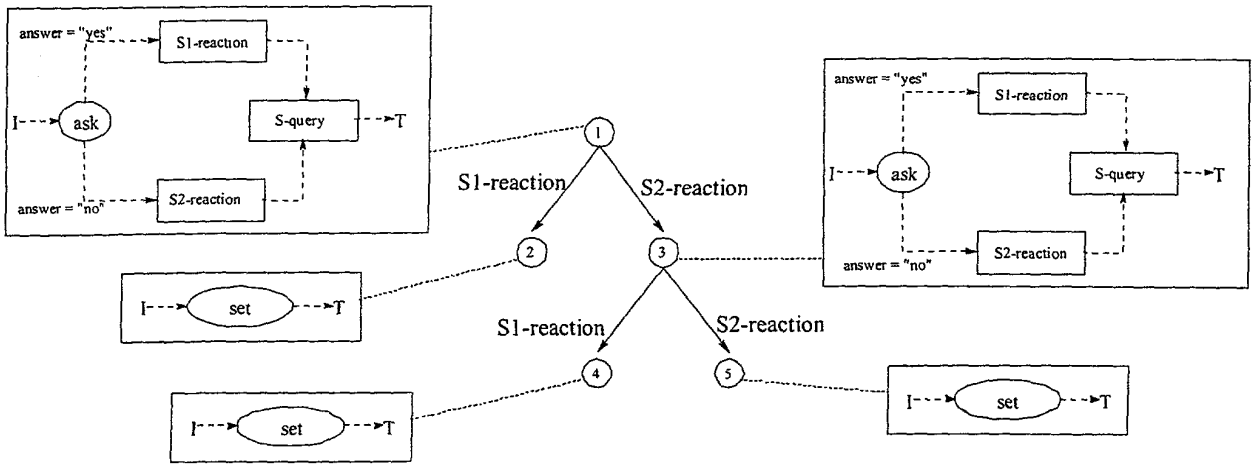

Fig. 7

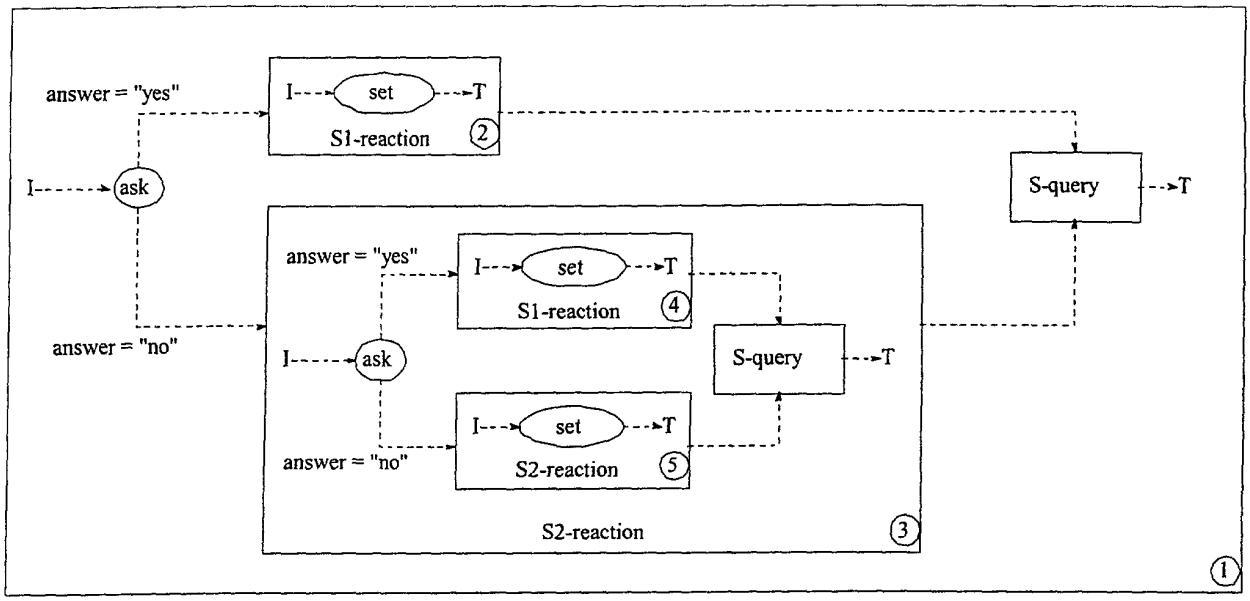
answer = "yes"
I
set
T
S1-reaction
2
I
ask
S-query
T
answer = "yes"
I
set
T
S1-reaction
4
answer = "no"
I
ask
S-query
T
I
set
T
answer = "no"
S2-reaction
5
S2-reaction
3
1

Fig. 8

```
<!element module (derived*, expression?, state*, module*>
                        <!attlist module          name CDATA #REQUIRED
                        number CDATA "1">
<!element derived (argument*, expression)>
<!attlist derived name CDATA>
<!element argument EMPTY>
<!attlist argument name CDATA>
<!element state (action*, transition*)>
<!attlist state name CDATA>
<!element transition (expression, path)>
<!element path (component?)>
<!attlist path state CDATA "initial">
<!element component (component?)>
                        <!attlist component      name CDATA #REQUIRED
                              number CDATA "1">
<!element expression (path | self | src | trg | evalattr | getfirst |
                        getnext | parent | root | apply | external | constant>
<!element action (setattr | ifthen | forall | external)>
<!element src EMPTY>
<!element trg EMPTY>
<!element self EMPTY>
<!element evalattr (expression?)>
<!attlist evalattr attribute CDATA #REQUIRED>
<!element getfirst (expression?)>
<!attlist getfirst attribute CDATA #REQUIRED>
<!element getnext (expression?)>
<!element parent (expression?)>
<!element root EMPTY>
<!element apply (expression, expression?)>
<!attlist apply op CDATA #REQUIRED>
<!element external (expression*)>
<!attlist external name CDATA
                language CDATA >
<!element constant EMPTY>
<!attlist constant value CDATA #REQUIRED>
<!element setAttr (expression?, expression)>
<!attlist setAttr attribute CDATA #REQUIRED>
<!element ifthenelse (expression, action*)>
<!element forall (action*)>
<!attlist forall range CDATA "all-elements"
                variable CDATA>
```

Fig. 9

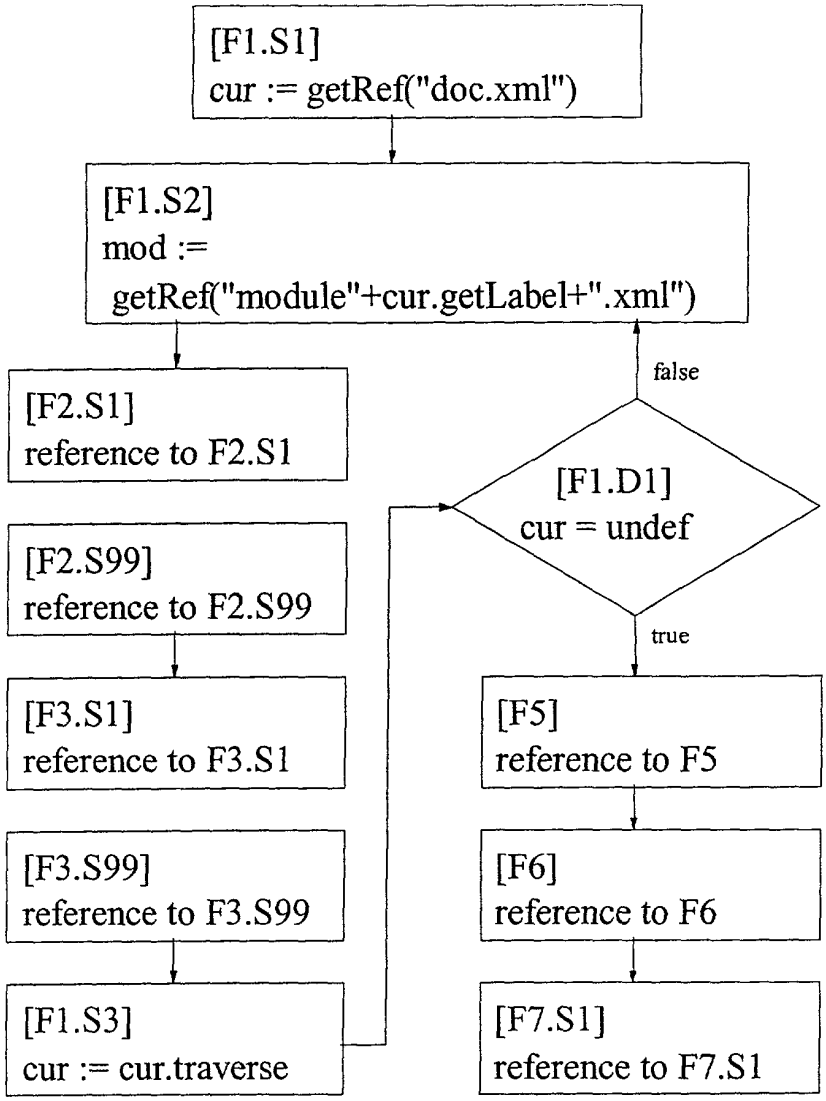
[F1.S1]
cur := getRef("doc.xml")
[F1.S2]
mod :=
getRef("module"+cur.getLabel+".xml")
false
[F1.D1]
cur = undef
true
[F2.S1]
reference to F2.S1
[F2.S99]
reference to F2.S99
[F3.S1]
reference to F3.S1
[F3.S99]
reference to F3.S99
[F1.S3]
cur := cur.traverse
[F5]
reference to F5
[F6]
reference to F6
[F7.S1]
reference to F7.S1

Fig. 10

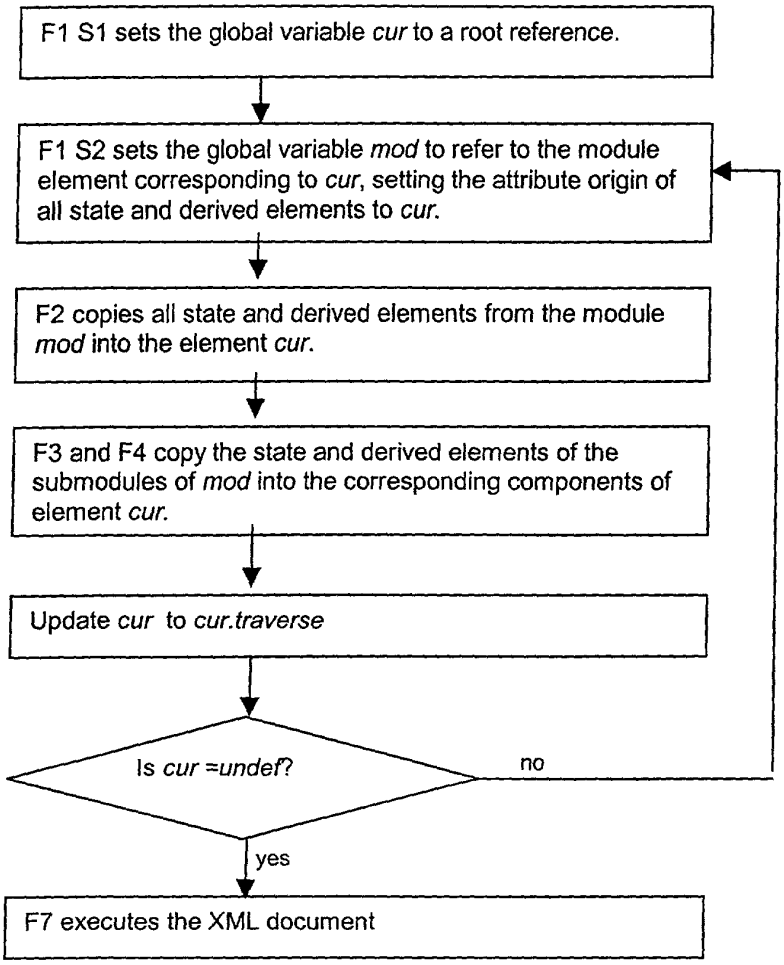
F1 S1 sets the global variable *cur* to a root reference.
F1 S2 sets the global variable *mod* to refer to the module element corresponding to *cur*, setting the attribute origin of all state and derived elements to *cur*.
F2 copies all state and derived elements from the module *mod* into the element *cur*.
F3 and F4 copy the state and derived elements of the submodules of *mod* into the corresponding components of element *cur*.
Update *cur* to *cur.traverse*
Is *cur* =*undef*?
no
yes
F7 executes the XML document

Fig. 10b

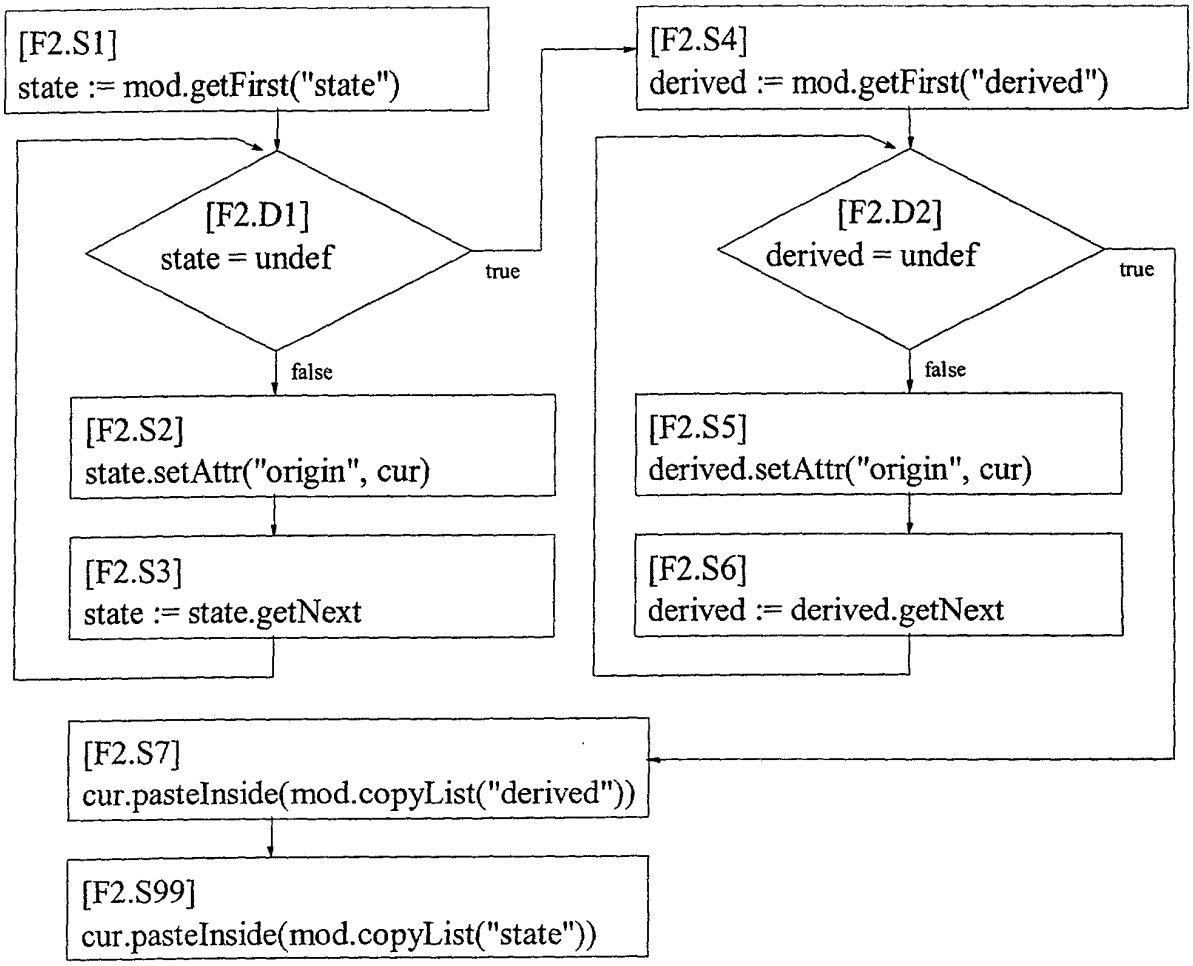
[F2.S1]
state := mod.getFirst("state")
[F2.D1]
state = undef
true
false
[F2.S2]
state.setAttr("origin", cur)
[F2.S3]
state := state.getNext
[F2.S4]
derived := mod.getFirst("derived")
[F2.D2]
derived = undef
true
false
[F2.S5]
derived.setAttr("origin", cur)
[F2.S6]
derived := derived.getNext
[F2.S7]
cur.pasteInside(mod.copyList("derived"))
[F2.S99]
cur.pasteInside(mod.copyList("state"))

Fig. 11

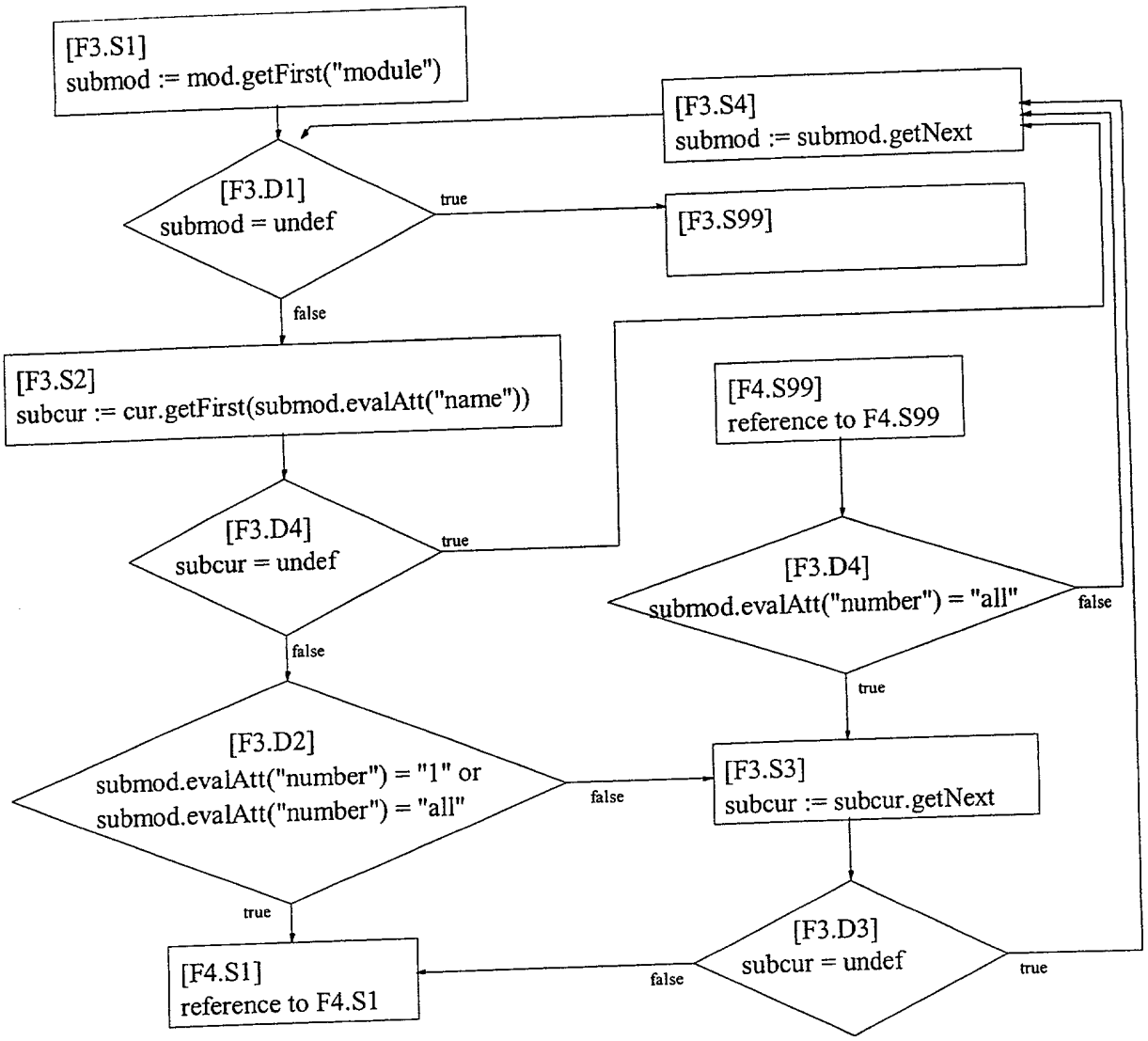
[F3.S1]
submod := mod.getFirst("module")
[F3.S4]
submod := submod.getNext
[F3.D1]
submod = undef
true
[F3.S99]
false
[F3.S2]
subcur := cur.getFirst(submod.evalAtt("name"))
[F4.S99]
reference to F4.S99
[F3.D4]
subcur = undef
true
[F3.D4]
submod.evalAtt("number") = "all"
false
false
true
[F3.D2]
submod.evalAtt("number") = "1" or
submod.evalAtt("number") = "all"
false
[F3.S3]
subcur := subcur.getNext
true
[F3.D3]
subcur = undef
true
[F4.S1]
reference to F4.S1
false

Fig. 12

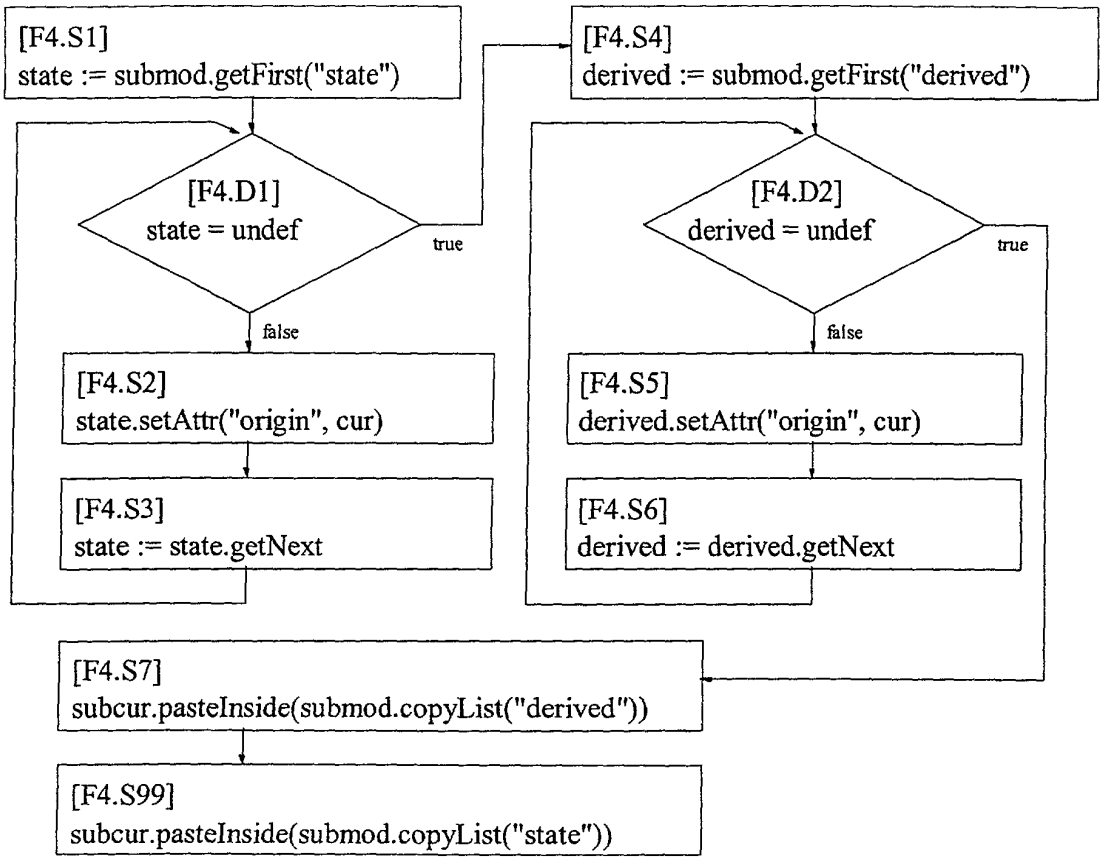
[F4.S1]
state := submod.getFirst("state")
[F4.D1]
state = undef
true
false
[F4.S2]
state.setAttr("origin", cur)
[F4.S3]
state := state.getNext
[F4.S4]
derived := submod.getFirst("derived")
[F4.D2]
derived = undef
true
false
[F4.S5]
derived.setAttr("origin", cur)
[F4.S6]
derived := derived.getNext
[F4.S7]
subcur.pasteInside(submod.copyList("derived"))
[F4.S99]
subcur.pasteInside(submod.copyList("state"))

Fig. 13

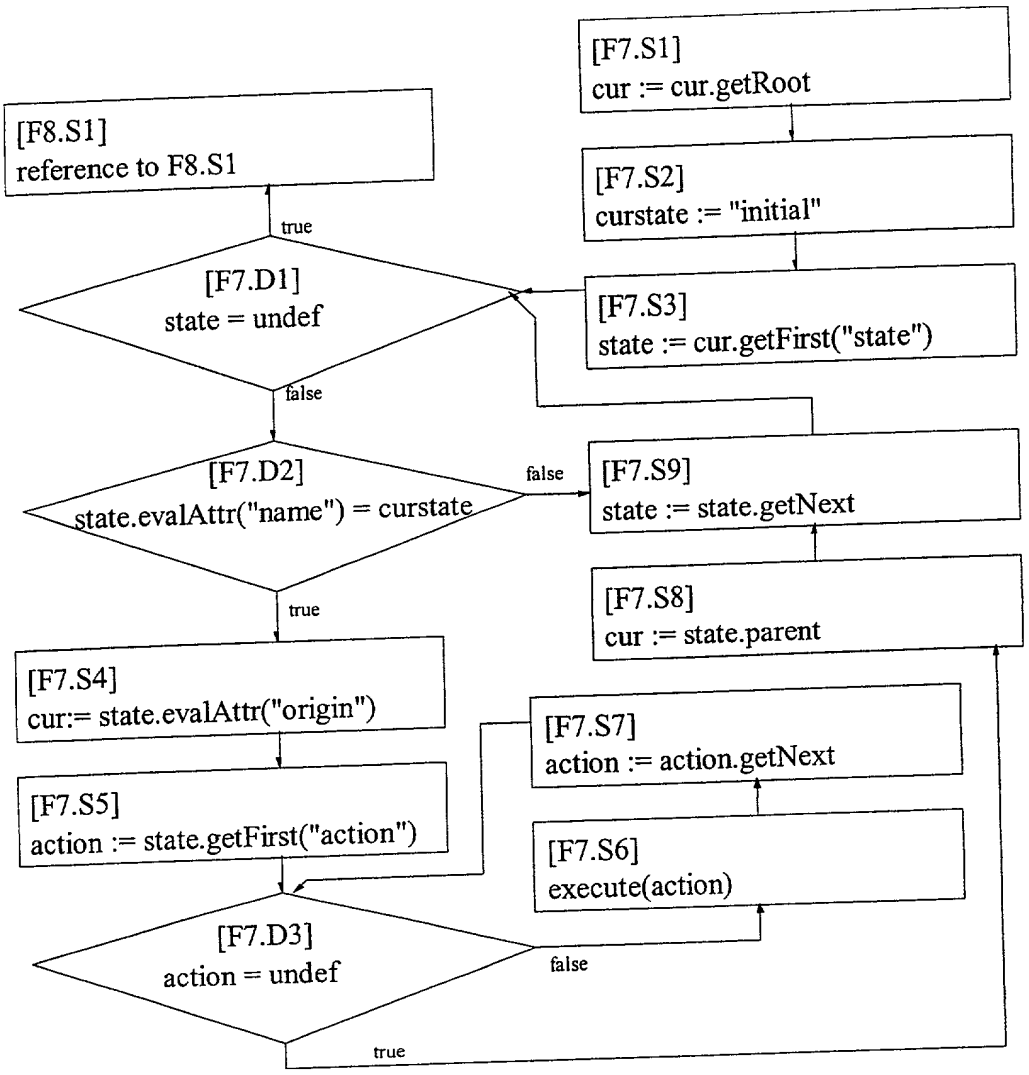
[F7.S1]
cur := cur.getRoot
[F7.S2]
curstate := "initial"
[F7.S3]
state := cur.getFirst("state")
[F7.D1]
state = undef
true
false
[F8.S1]
reference to F8.S1
[F7.D2]
state.evalAttr("name") = curstate
false
true
[F7.S9]
state := state.getNext
[F7.S8]
cur := state.parent
[F7.S4]
cur:= state.evalAttr("origin")
[F7.S5]
action := state.getFirst("action")
[F7.S7]
action := action.getNext
[F7.S6]
execute(action)
[F7.D3]
action = undef
false
true

Fig. 14

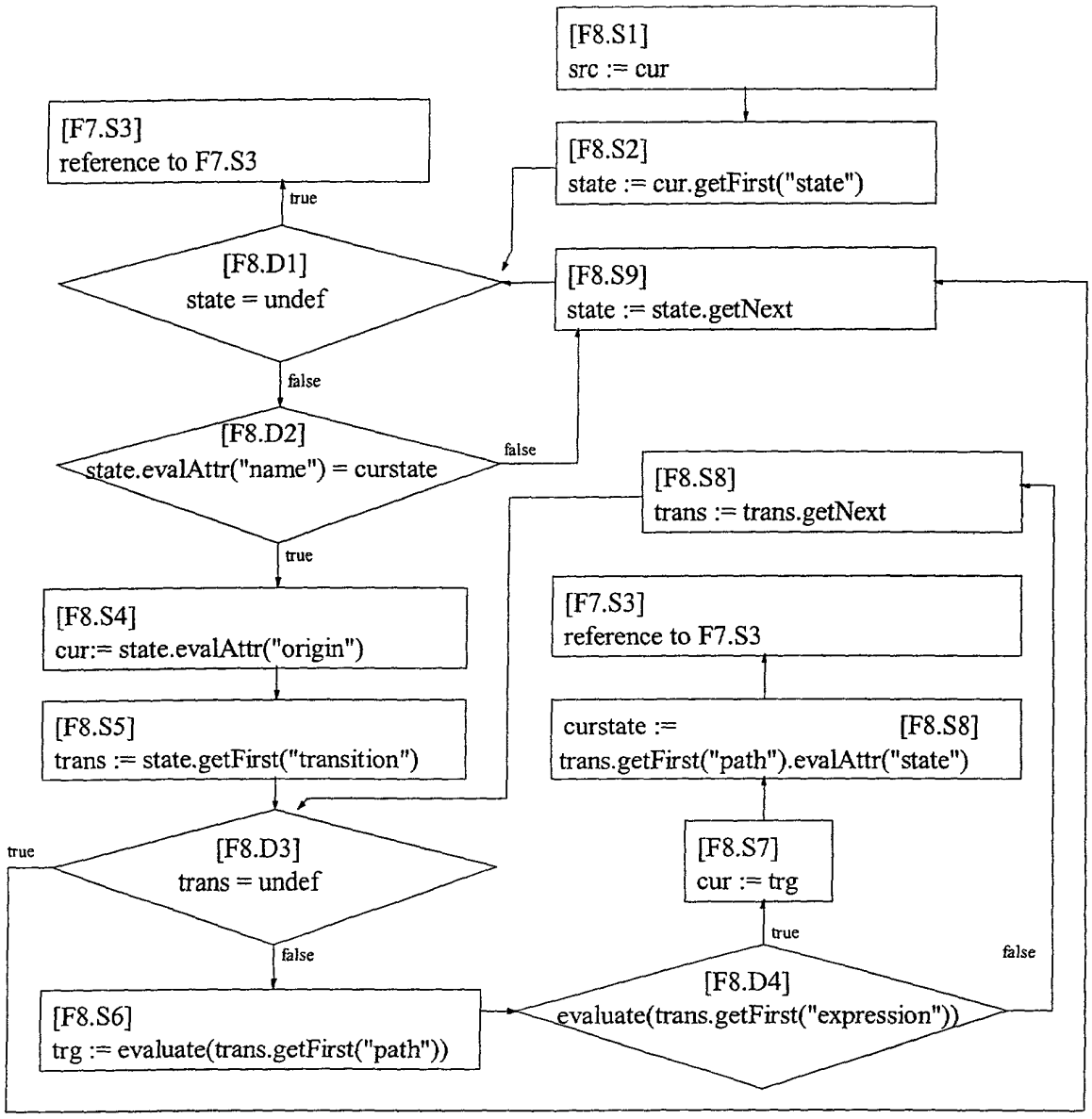
[F8.S1]
src := cur
[F8.S2]
state := cur.getFirst("state")
[F7.S3]
reference to F7.S3
true
[F8.D1]
state = undef
[F8.S9]
state := state.getNext
false
[F8.D2]
state.evalAttr("name") = curstate
false
[F8.S8]
trans := trans.getNext
true
[F8.S4]
cur:= state.evalAttr("origin")
[F7.S3]
reference to F7.S3
[F8.S5]
trans := state.getFirst("transition")
curstate := [F8.S8]
trans.getFirst("path").evalAttr("state")
true
[F8.D3]
trans = undef
[F8.S7]
cur := trg
false
true
false
[F8.S6]
trg := evaluate(trans.getFirst("path"))
[F8.D4]
evaluate(trans.getFirst("expression"))

Fig. 15

```
modulequery.xml:
<module name="query">
        <state  name="initial">
                <transition> <path state="ask"/> </transition>
        </state>
        <state  name="ask">
                <action>
                        <external  name="print" language="shell">
                                <expression> <attreval  name="question"/> </expression>
                        </external>
                </action>
                <action>
                        <setattr name="answer">
                                <expression> <self/> </expression>
                                <expression>
                                        <external name="get(stdin)" language="shell"> </external>
                                </expression>
                        </setattr>
                </action>
                <transition>
                        <expression>
                                <apply op="=">
                                        <expression> <attreval name="answer"/> </expression>
                                        <expression> <constant value="yes"/> </expression>
                                </apply>
                        </expression>
                        <path>  <component  name="reaction" number="1"/> </path>
                </transition>
                <transition>
                        <expression>
                                <apply op="=">
                                        <expression> <attreval name="answer"/> </expression>
                                        <expression> <constant value="no"/> </expression>
                                </apply>
                        </expression>
                        <path>  <component  name="reaction" number="2"/> </path>
                </transition>
        <module name="action"  number="1">
                <state  name="terminal">
                        <transition> <path> <component  name="query"> </path> </transition>
                </state>
        </module>
        <module name="action"  number="2">
                <state  name="terminal">
                        <transition><path> <component  name="query"> </path></transition>
                </state>
        </module>
        <module  name="query">
                <state  name="terminal">
                        <transition> <path name="terminal"/> </transition>
                </state>
        </module>
</module>
```

Fig. 16

```
module reaction.xml:
<module name="reaction">
        <state name="initial">
                <transition>
                        <expression>
                                <apply op="!=">
                                        <expression> <getfirst name="setpoints"/> </expression>
                                        <expression> <constant value="undef"> </expression>
                                </apply>
                        </expression>
                        <path> <component name="setpoints"/> </path>
                </transition>
                <transition>
                        <expression>
                                <apply op="!=">
                                        <expression> <getfirst name="query"/> </expression>
                                        <expression> <constant value="undef"> </expression>
                                </apply>
                        </expression>
                        <path> <component name="query"/> </path>
                </transition>
        </state>
        <module name="setpoints">
                <state name="terminal"> <transition> <path  state="terminal"/> </transition> </state>
        </module>
        <module name="query">
                <state name="terminal"> <transition> <path  state="terminal"/> </transition> </state>
        </module>
</module>
```

Fig. 17

```
module setpoints.xml:
<module name="setpoints">
        <state  name="initial">
                <transition> <path state="set"/> </transition>
        </state>
        <state  name="set">
                <action>
                        <setattr name="PointCounter">
                                <expression> <getroot/> </expression>
                                <expression>
                                        <apply op="+">
                                                <expression>
                                                        <getattr name="PointCounter">
                                                                <expression> <getroot/> </expression>
                                                        </getattr>
                                                </expression>
                                                <expression> <getattr name="points"/> </expression>
                                        </apply>
                                </expression>
                        </setattr>
                </action>
                <transition> <path state="terminal"/> </transition>
        </state>
</module>
```

Fig. 18

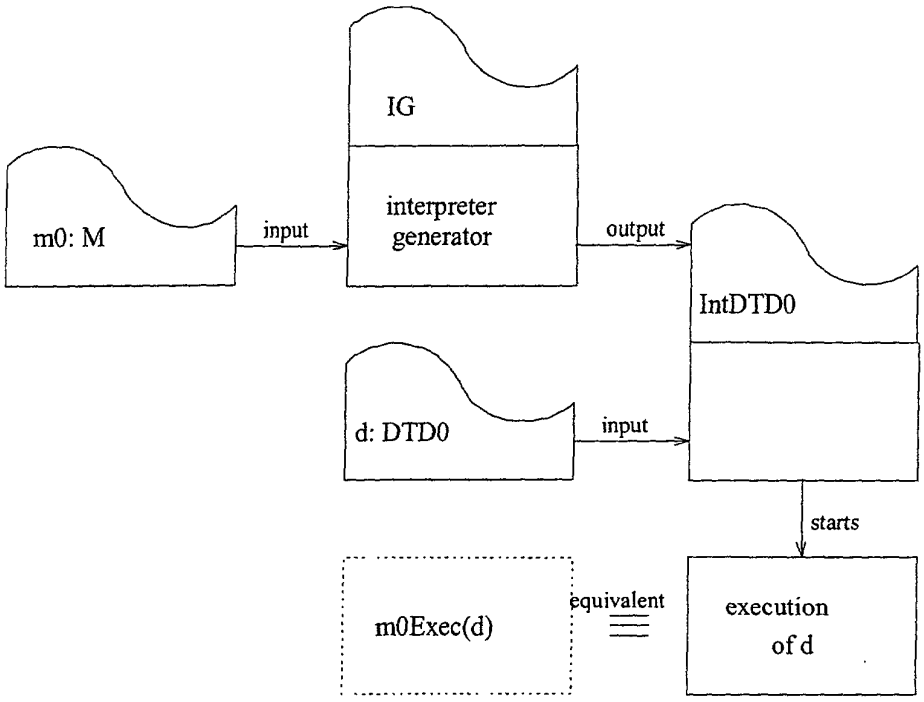

Fig. 19a

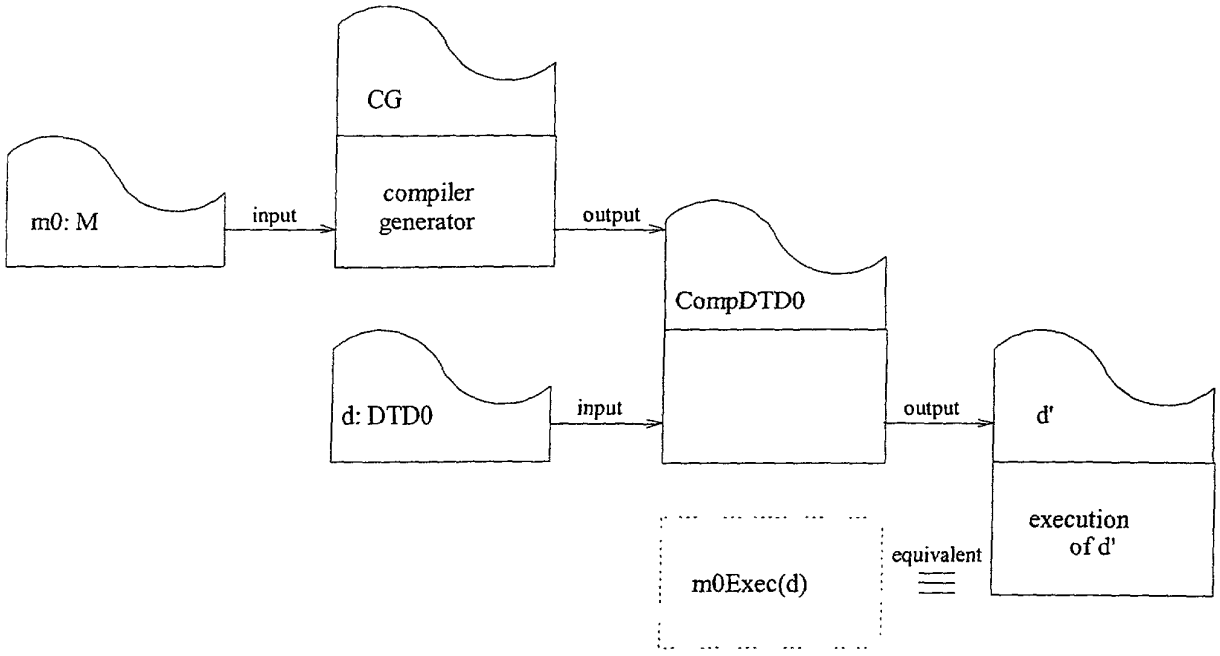

Fig. 19b

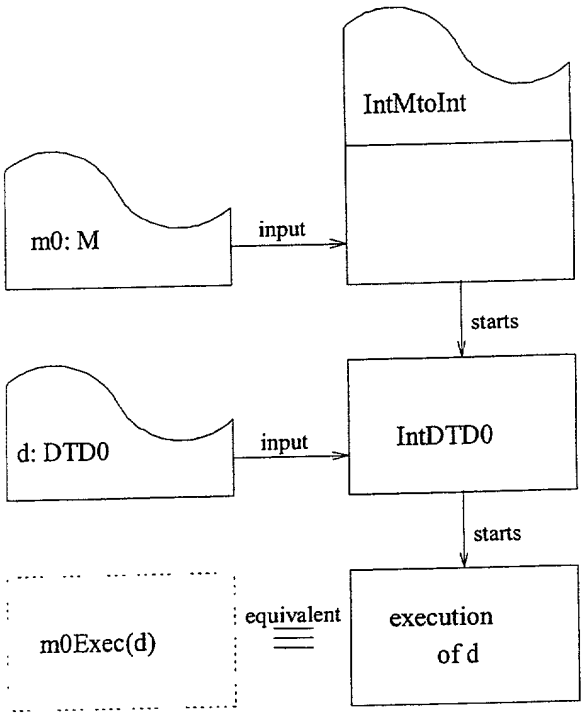

Fig. 19c

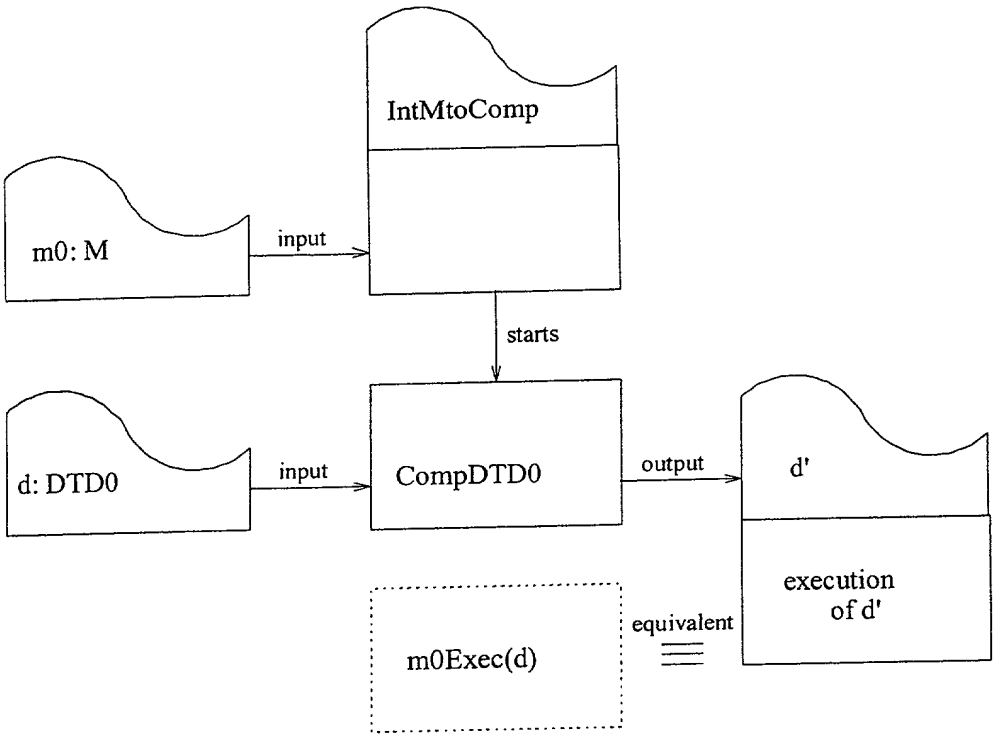

Fig. 19d

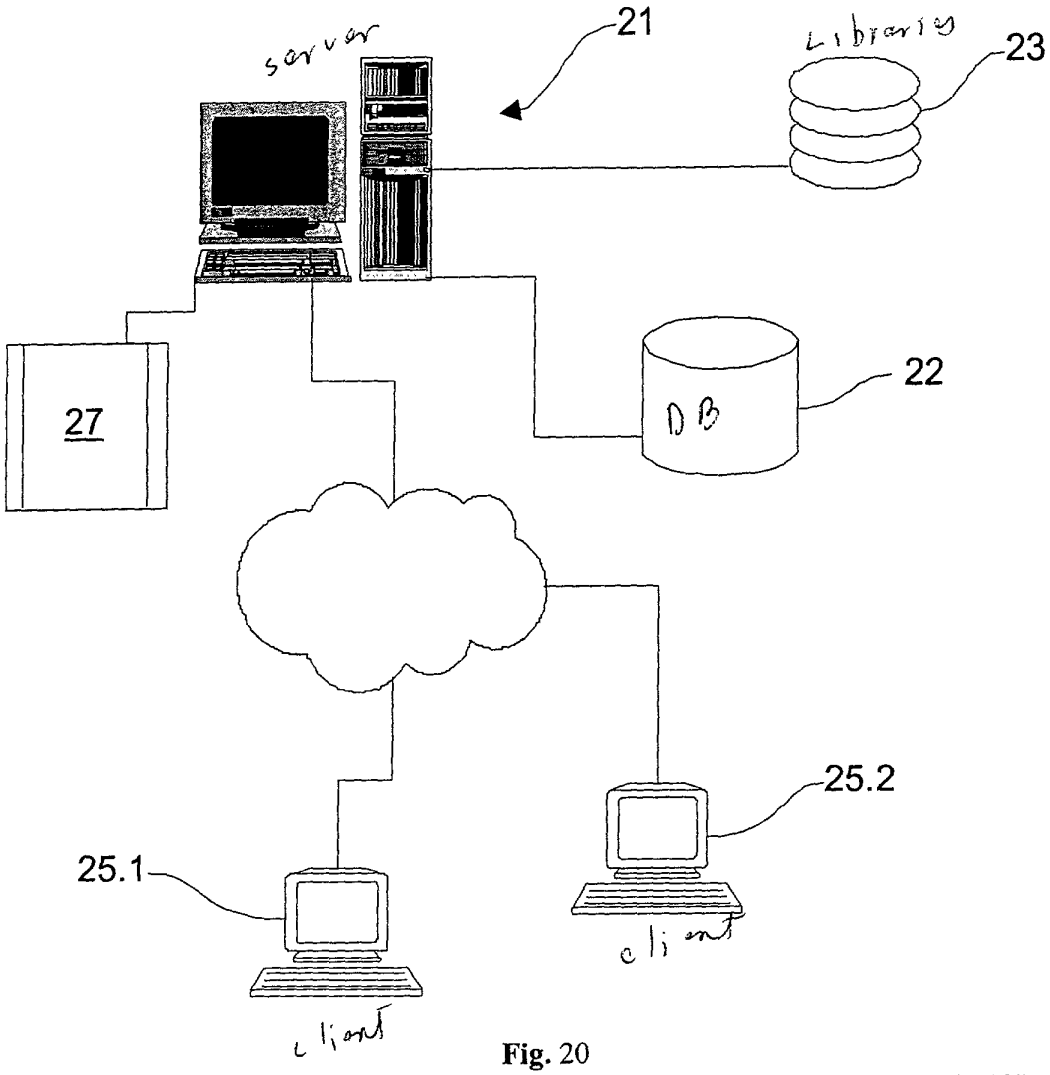

Fig. 20